

Stream Cipher

메시지를 암호화하는 키의 bit Stream처럼 축출된다.

Jong Hwan Park

Bit-oriented Cipher

- Character-oriented cipher
 - Classical symmetric-key cipher
 - Information is only in text type
- Bit-oriented cipher
 - Modern symmetric-key cipher
 - Information consists of various types of texts, numbers, graphics, audio, and video data
 - Need *encoding* method to convert those data into bits (01100···)
 - ASCII code:
 - m → 01101101
 - d → 01100100
 - n → 01101110

XOR Property (1)

- XOR operation in computer

$$0 \text{ XOR } 0 = 0$$

$$0 \text{ XOR } 1 = 1 \quad \Rightarrow \quad (\text{mod } 2)$$

$$1 \text{ XOR } 0 = 1$$

$$1 \text{ XOR } 1 = 0$$

- $01001100 \dots A$

$$\oplus \ 10101010 \dots B$$

$$11100110 \dots C = A \oplus B$$

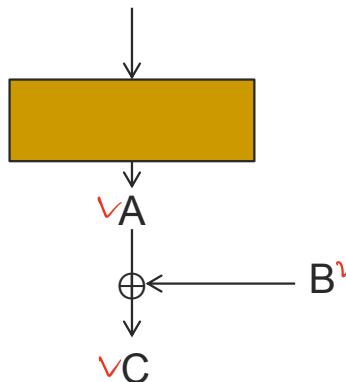
XOR Property (2)

- Cancelation (or masking) effect of XOR

$$\begin{array}{r} 11100110 \\ \oplus \quad 10101010 \\ \hline \end{array} \quad \begin{array}{l} A \oplus B = C \\ \text{message} \\ \text{key} \end{array}$$

$$01001100 \quad C \oplus B = A \oplus B \oplus B = A$$

- XOR operation in algorithm

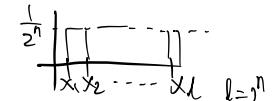
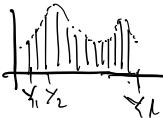


$$\boxed{A \oplus B = C}$$
$$A \oplus C = B$$
$$B \oplus C = A$$

XOR Property (3)

$$x \in \{0,1\}^n \Rightarrow x = \overbrace{1010 \dots 110}^{n \text{개}}$$

Thm: Y: a variable over $\{0,1\}^n$, X: an independently uniformly random variable on $\{0,1\}^n$



Then $Z := \underline{\underline{Y}} \oplus X$ is uniform variable on $\{0,1\}^n$

입력변수
랜덤변수

\Rightarrow 평균에 대한 통계 Y는 다 사라지고 정상이 되었지만 X는 Uniform한 통계로 바뀐다. \Rightarrow X의 통계를 봅니다.

Proof: (for $n=1$)

$$\Pr[Z=0] = \Pr[(x,y) = (0,0) \text{ or } (x,y) = (1,1)] =$$

$\Pr[Z=0 \cap Y=0]$ 두 가지 경우에 대한 합집합

$$= \Pr[(x,y) = (0,0)] + \Pr[(x,y) = (1,1)] =$$

$$= \frac{P_0}{2} + \frac{P_1}{2} = \frac{1}{2}$$

$$\Pr[Z=1] = \Pr[(x,y) = (1,0) \text{ or } (x,y) = (0,1)] =$$

$$= \Pr[(x,y) = (1,0)] + \Pr[(x,y) = (0,1)]$$

$$= \frac{P_0}{2} + \frac{P_1}{2} = \frac{1}{2} (\because P_0 + P_1 = 1)$$

$\therefore X$ 를 선택할 확률이 0과 1

$$\Pr[Z=0 | Y=0] = \frac{\Pr[Z=0 \cap Y=0]}{\Pr[Y=0]} = \frac{\Pr[Z=0]}{\Pr[Y=0]} = \frac{P_0}{P_0} = \frac{1}{2}$$

$$\Pr[Z=0 | Y=1] = \frac{\Pr[Z=0 \cap Y=1]}{\Pr[Y=1]} = \frac{\Pr[Z=0]}{\Pr[Y=1]} = \frac{P_0}{P_1} = \frac{1}{2}$$

$\Rightarrow Z=0$ 은 흔히 가는 확률이 0이지

\Rightarrow 1은 흔히 가는 확률이 0이지
알수있다. 확률이 같을 때 확률은 같아야 한다.

Y	Pr
0	P_0
1	P_1

X	Pr
0	$1/2$
1	$1/2$

X	Y	Pr
0	0	$P_0/2$
0	1	$P_1/2$
1	0	$P_0/2$
1	1	$P_1/2$

X	Y	Pr
0	0	$P_0/2$
0	1	$P_1/2$
1	0	$P_0/2$
1	1	$P_1/2$

\leftarrow X가 정의되는

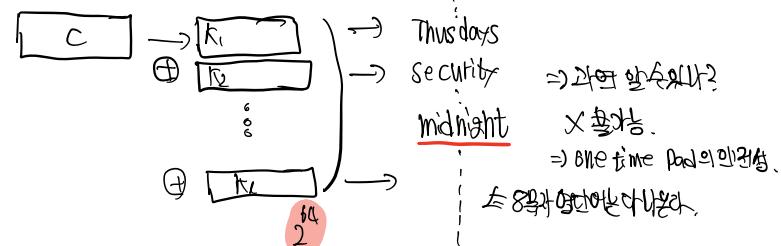
\leftarrow Y가 정의되는 확률이 같다.
 \leftarrow 1과 0의 확률이 같다.

$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$

\leftarrow 확률이 같은 경우에
평균을 정의하는게 가능하다.

One-Time Pad (Vernam's Cipher) (1)

- Vernam presented the one-time pad in 1917
 - MS, KS, CS are all $\{0, 1\}^n$
 - Gen (1^n) \rightarrow a random string k in KS (with uniform distribution)
 - Enc (k, m) \rightarrow $c = \underline{k} \oplus \underline{m}$
 - Dec (k, c) \rightarrow $m = \underline{k} \oplus \underline{c}$



- ## ■ Encryption of m=midnight

Key 01101011 11111010 01001000 11011000 01100101 11010101 10101111 00011100

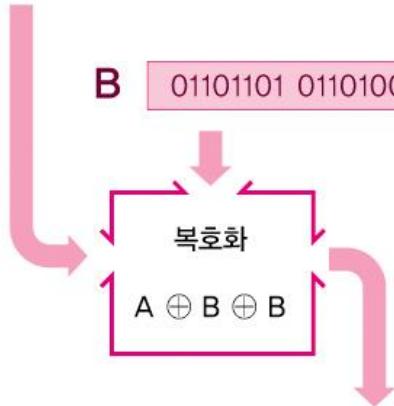
C 00000110 10010011 00101100 10110110 00001100 10110010 11000111 01101000

One-Time Pad (Vernam's Cipher) (2)

■ Decryption of the ciphertext

A \oplus B 01101101 01101001 01100100 01101110 01101001 01100111 01101000 01110100

B 01101101 01101001 01100100 01101110 01101001 01100111 01101000 01110100



A 01101101 01101001 01100100 01101110 01101001 01100111 01101000 01110100



One-Time Pad (Vernam's Cipher) (3)

- One-time pad is perfectly secure 그대로 풀수있다.
 - An adversary who intercepts a ciphertext learns absolutely nothing about the plaintext (\rightarrow against ciphertext-only attack)
 - Even an adversary with unbounded computational power cannot learn any information about the plaintext
 - Security does not rely on any of unproven assumptions
 - OTP is the only perfectly secure encryption in cryptography

■ Why?

- For a randomly chosen key with uniform distribution, we have

$$\Pr[\text{C} = c \mid m = m_0] = \frac{1}{2^n} = \Pr[\text{C} = c \mid m = m_1]$$

$$P_f[x=0 \mid Y=0] = \frac{1}{2}$$

$$P_f[x=0 \mid Y=1] = \frac{1}{2}$$

즉 확률은 동일한 경우는 2개
= 흑정암호문이 흰암 풍향에 대해 $= \frac{1}{2^n}$ 3 확률이다.

One-Time Pad (Vernam's Cipher) (4)

■ Drawbacks of one-time pad

- The key is required to be as long as the message ($|k|=|m|=|c|$)
- Only secure if used once with the same key

■ Why?

two time pad

Using the same key k , $c_1 = m_1 \oplus k$ and $c_2 = m_2 \oplus k$

Then, an adversary has that $c_1 \oplus c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$

Enough redundancy in English and ASCII encoding that:

$$m_1 \oplus m_2 \rightarrow m_1, m_2 \quad m_1, m_2 \text{의 } \oplus \text{은 } X. \text{ } \Rightarrow \text{two time pad } \text{은 } \text{복호화가 } \text{쉽다.}$$

\Rightarrow two time pad 복호화 (복호화할 때 같은 키를 사용함)

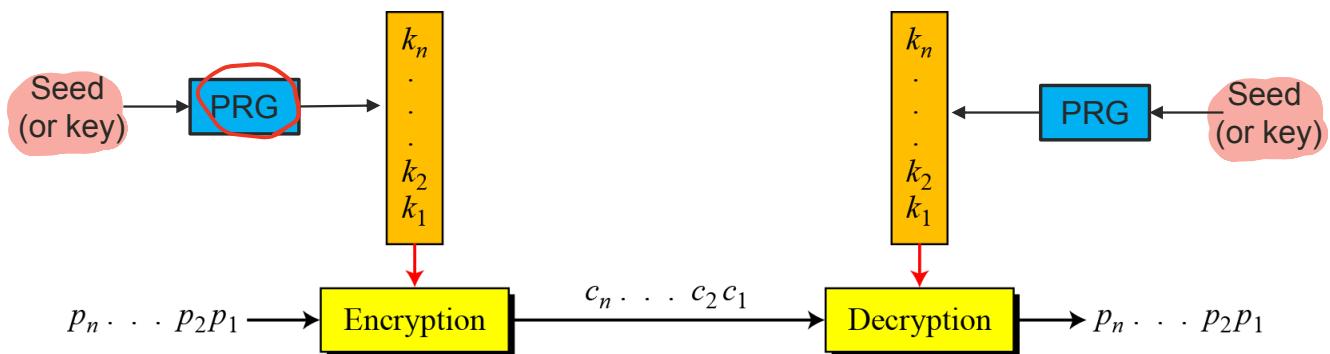
○ Key distribution & key sync.

■ Never can be used ?

- Project Venona
- Hot line/

Stream Cipher

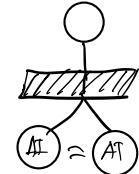
- Idea of stream cipher
 - Inspired by one-time pad



- Key stream is generated by using a pseudorandom generator (PRG)
↑ All the random X
- PRG(short key=seed) → long key stream
- Encryption is simply XORed with $\text{PRG}(k) \oplus m$ (very fast)
- Need the short key shared in advance
- Security depends on how good key streams look ‘random’

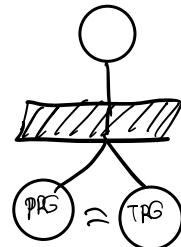
Security of PRGs

- Indistinguishability between PRG and TRG streams
 - PRG – Pseudo-Random Generator
 - TRG – True Random Generator



- Randomness check tests for PRG

인정성.

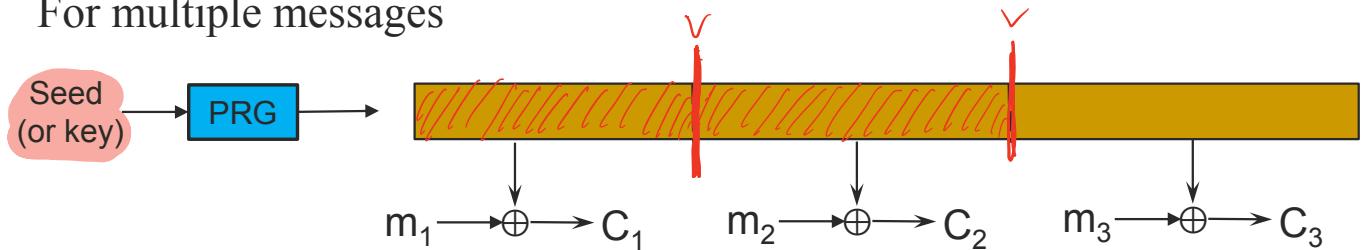


Stream Cipher Modes of Operation

2) PRG

■ Synchronized mode

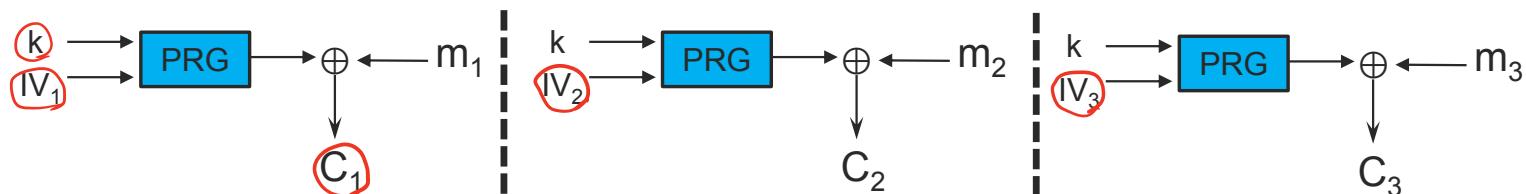
- Encryption is done by $C = \text{PRG}(k) \oplus m$
- For multiple messages



- Need to maintain synchronized state

■ Unsynchronized mode (stateless way)

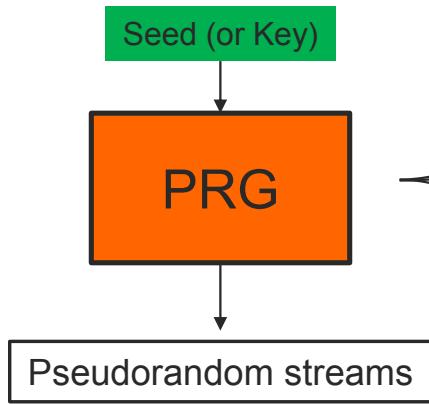
- Encryption is done by $C = \text{PRG}(k, IV) \oplus m$ for initialization vector
- For multiple messages



Several Approaches to PRGs

PRG들을 한데 모으는 방식.

3가지



- LFSR-based
- Dedicated to stream cipher
 - RC4, and Salsa, Trivium, and others
(from sStream project in Europe)
- Block cipher-based(later)
 - CTR, OFB mode

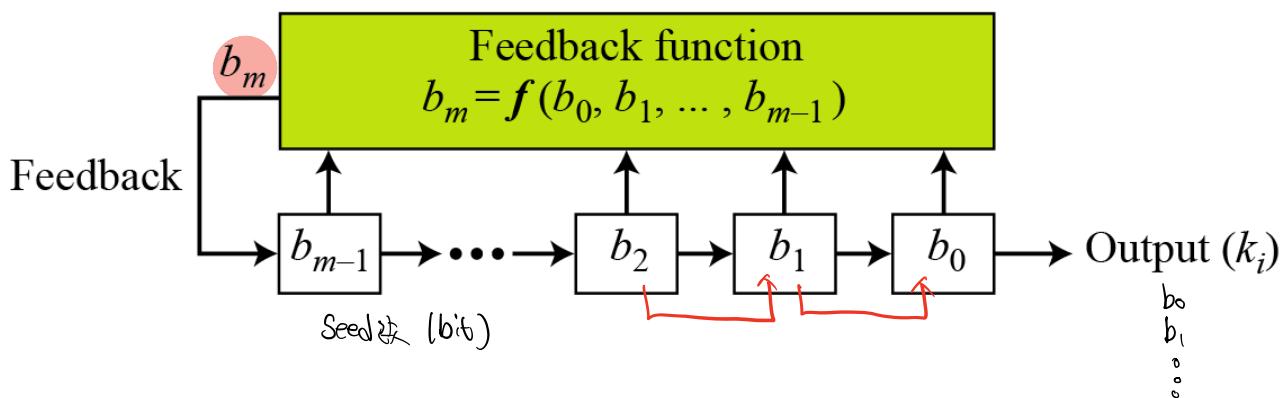
block cipher 대체로 인식한
block cipher을 가지고 페어링되어
Mode라고.
-> CTR은 Stream cipher 품목에 넣어야 한다.

Feedback Shift Register

- Feedback Shift Register (FSR)
 - Seed = $(b_{m-1}, \dots, b_1, b_0)$ as a secret key

Transition

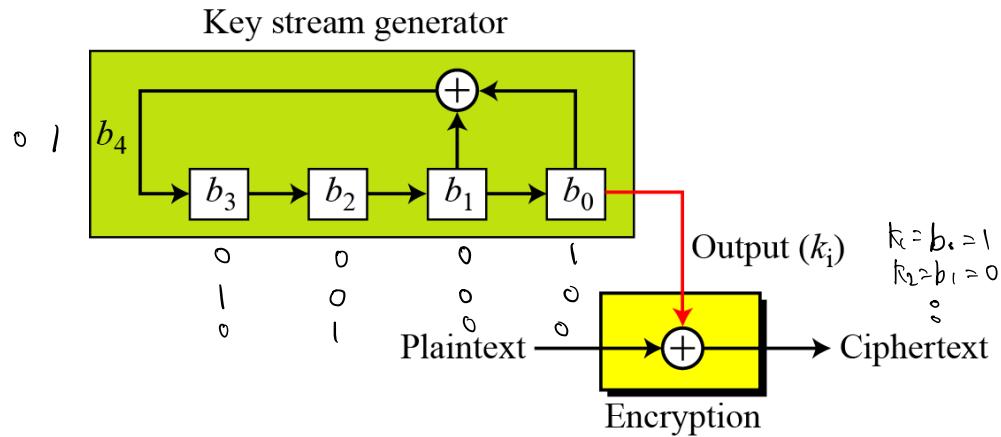
b_0	\rightarrow	k_i
b_1	\rightarrow	b_0
b_2	\rightarrow	b_1
\dots		
b_m	\rightarrow	b_{m-1}



LFSR for PRG

Linear Feedback Shift Register (LFSR)

- Seed = $(b_{m-1}, \dots, b_1, b_0)$
- $b_m = c_{m-1}b_{m-1} \oplus \dots \oplus c_2b_2 \oplus c_1b_1 \oplus c_0b_0$ feedback function.
- Example of LFSR where $b_4 = b_1 \oplus b_0$ if the seed = $(0001)_2$



- Show the value of output for 20 transitions (shifts)

Outputs of LFSR (1)

States	b_4	b_3	b_2	b_1	b_0	k_i
Initial	1	0	0	0	1	
1	0	1	0	0	0	1
2	0	0	1	0	0	0
3	1	0	0	1	0	0
4	1	1	0	0	1	0
5	0	1	1	0	0	1
6	1	0	1	1	0	0
7	0	1	0	1	1	0
8	1	0	1	0	1	1
9	1	1	0	1	0	1
10	1	1	1	0	1	0

bit₂

Outputs of LFSR (2)

(Continued)

11	1	1	1	1	0	1
12	0	1	1	1	1	0
13	0	0	1	1	1	1
14	0	0	0	1	1	1
15	1	0	0	0	1	1
16	0	1	0	0	0	1
17	0	0	1	0	0	0
18	1	0	0	1	0	0
19	1	1	0	0	1	0
20	1	1	1	0	0	1

$$\text{정답의 수} = 2^4 - 1 = 2^4 - \text{임의 } 10000$$

Outputs of LFSR (3)

- The key stream is

100010011010111 | 100010011010111 | 100010011010111 | 100010011010111 ...

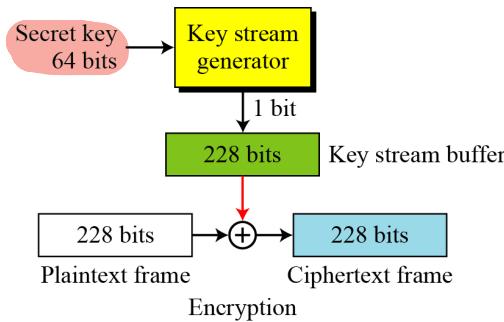
- Which looks like a random sequence
 - But it is a repetition of 15 bits
-
- The key stream from a LFSR with m-bit seed is repeated after N bits
 - Key stream less than N bits acts as a pseudorandom sequence
 - Maximum period $N = \underline{2^m - 1}$ (why?)

Seed 초기에 둘의 bit stream이 초기화되면 그 다음에는 동일한 bit stream이 나오는 경우에 있다.
= Seed 초기에 동일한 값이 위치하는 경우

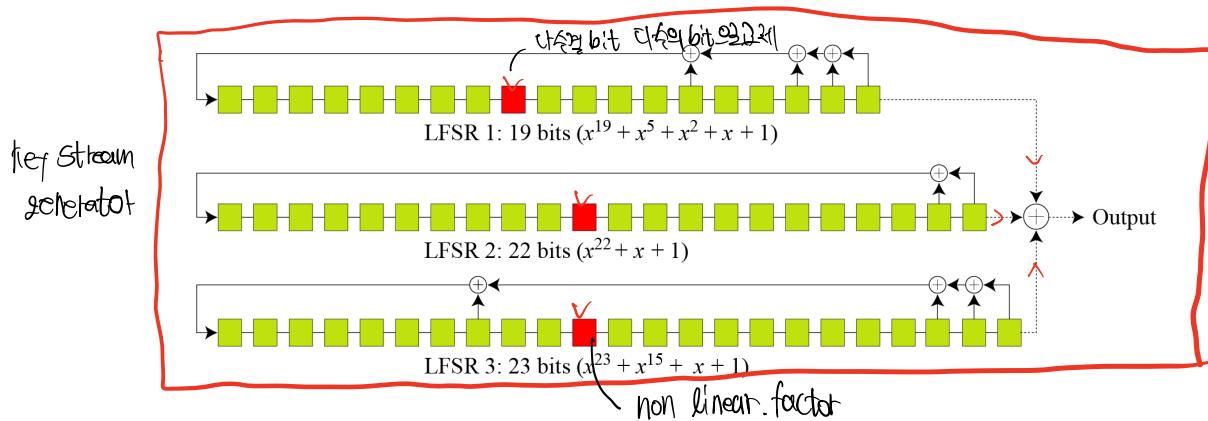
$2^m - 1$ 을 통과하면 같은 값이 나온다.

A5/1 using LFSRs

- Used in Global System for Mobile (GSM) communication
 - Security attacks were reported in 2000 and 2003
 - General outline:



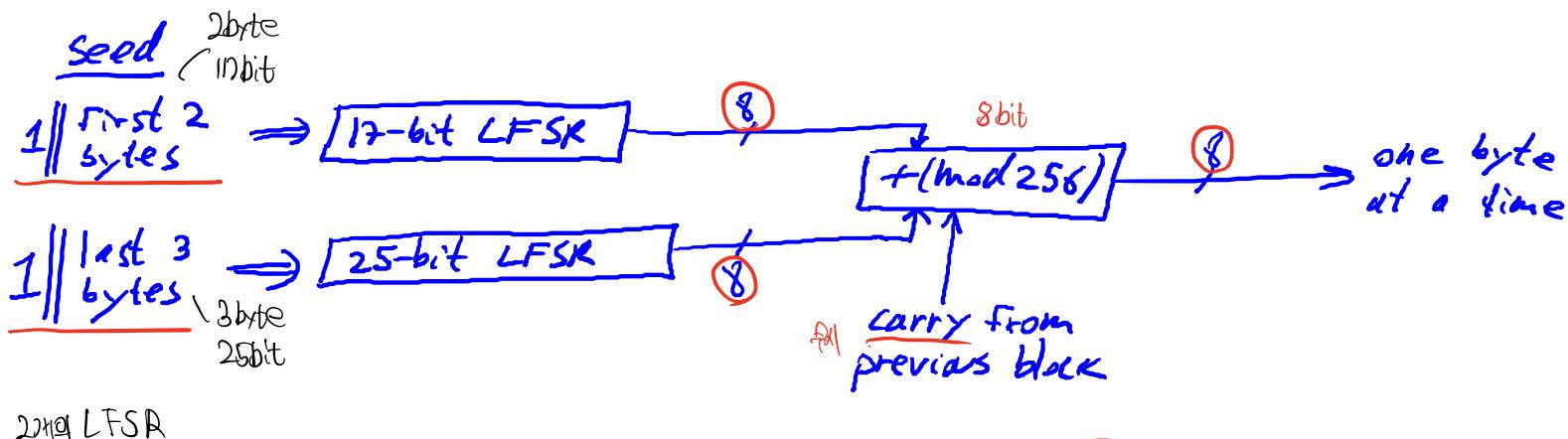
- Use 3 LFSRs with 19, 22, and 23 bits



☞ 사용해 나온다.

Old Example(hardware): CSS

- Used in DVD encryption: badly broken
 - http://en.wikipedia.org/wiki/Content_Scramble_System
- CSS (Content Scrambling System): seed = 5 bytes = 40 bits
40-bit security level
 2^{40}



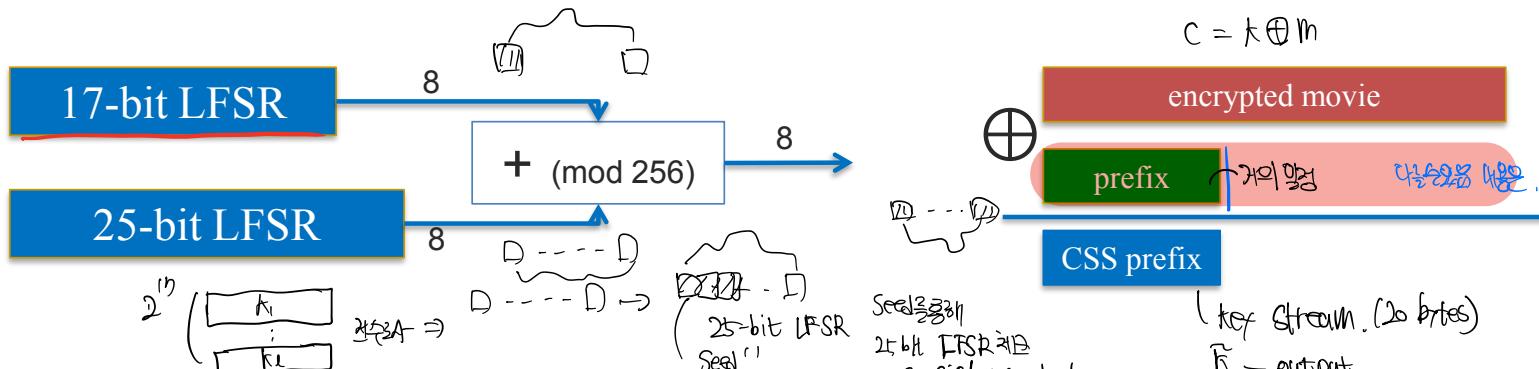
22nd LFSR

Easy to break in time $\approx 2^{17}$

☞ 시험에 나온다. 중요

Cryptanalysis of CSS

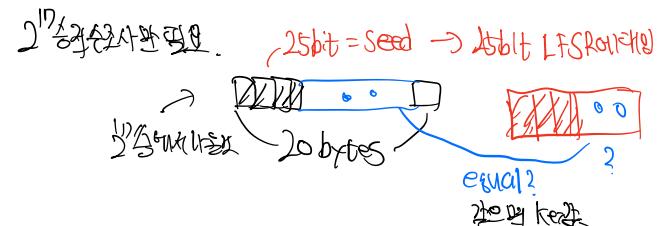
- 2^{17} time attack (using 2^{40} -bit key)



For all possible initial settings of 17-bit LFSR do:

- Run 17-bit LFSR to get 20 bytes of output
- Subtract from CSS prefix \Rightarrow candidate 20 bytes output of 25-bit LFSR
- If consistent with 25-bit LFSR, found correct initial settings of both !!

Using key, generate entire CSS output

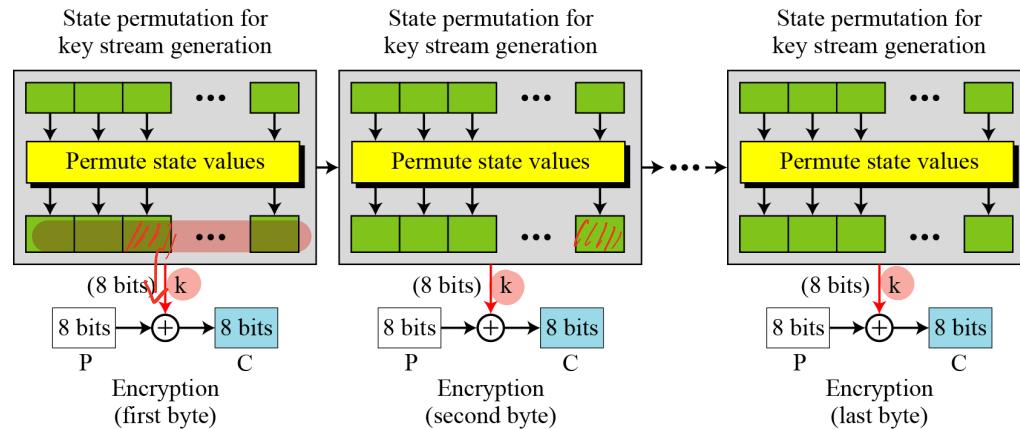
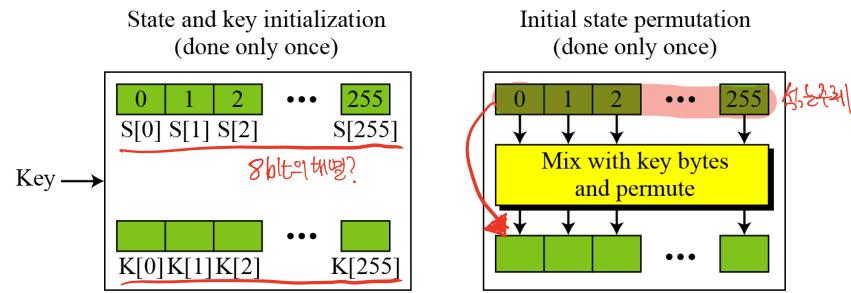


RC4(software)

■ RC4

사용자

- Designed in 1987 by Ronald Rivest for RSA Data Security
- Used in SSL/TLS and WEP encryption protocol in IEEE802.11 wireless networks
- Idea:



RC4 – Encryption Algorithm (1)

Algorithm 8.6 *Encryption algorithm for RC4*

RC4_Encryption (K)

{

// Creation of initial state and key bytes

for ($i = 0$ to 255)

{

$S[i] \leftarrow i$

$K[i] \leftarrow Key[i \bmod KeyLength]$

}

// Permuting state bytes based on values of key bytes

$j \leftarrow 0$

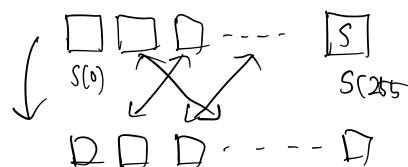
for ($i = 0$ to 255)

{

$j \leftarrow (j + S[i] + K[i]) \bmod 256$

swap ($S[i]$, $S[j]$)

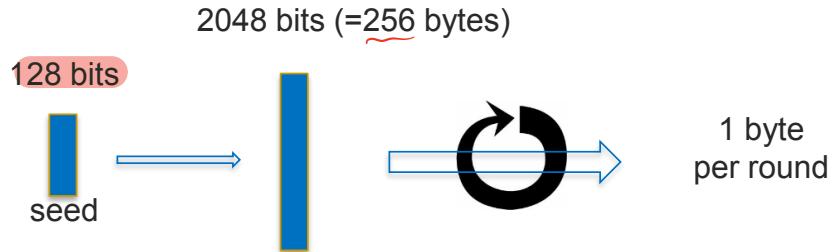
}



RC4 – Encryption Algorithm (2)

```
// Continuously permuting state bytes, generating keys, and encrypting
i ← 0
j ← 0
while (more byte to encrypt)
{
    i ← (i + 1) mod 256
    j ← (j + S[i]) mod 256
    swap (S [i] , S[j])
    k ← S [(S[i] + S[j]) mod 256]
    // Key is ready, encrypt
    input P
    C ← P ⊕ k
    output C
}
```

Cryptanalysis of RC4



- RC4($K=128$ -bit seed) \rightarrow randomly-looking key stream
- Weaknesses:
 - 1. Bias in initial output: $\Pr[\text{2nd byte} = 0] = 2/256$
 - 2. Prob. of $(0,0)$ is $\underbrace{1/256^2}_{\text{real}} + \underbrace{1/256^3}_{\text{统计偏差}}$
 - 3. Related key attacks

$$\text{prob random} = \frac{1}{256}$$

统计偏差。

Real Example of Stream Cipher (1)

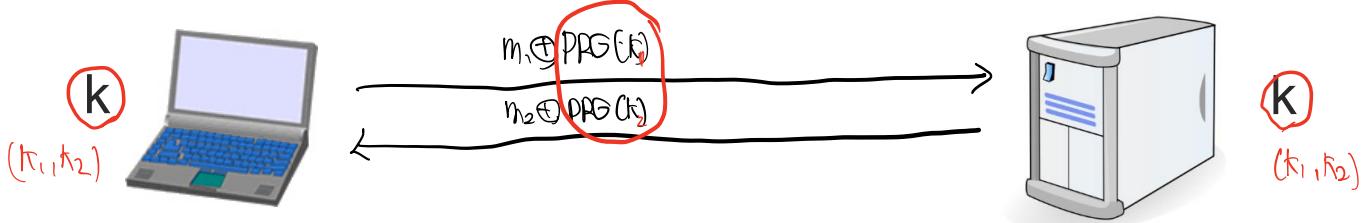
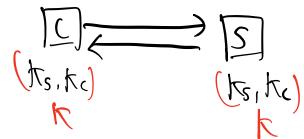
- As in OTP, never use stream cipher key more than once!!

$$C_1 \leftarrow m_1 \oplus \text{PRG}(k)$$

$$C_2 \leftarrow m_2 \oplus \text{PRG}(k)$$

Eavesdropper does: $C_1 \oplus C_2 \rightarrow m_1 \oplus m_2 \rightarrow m_1, m_2$

- MS-PPTP (windows NT):

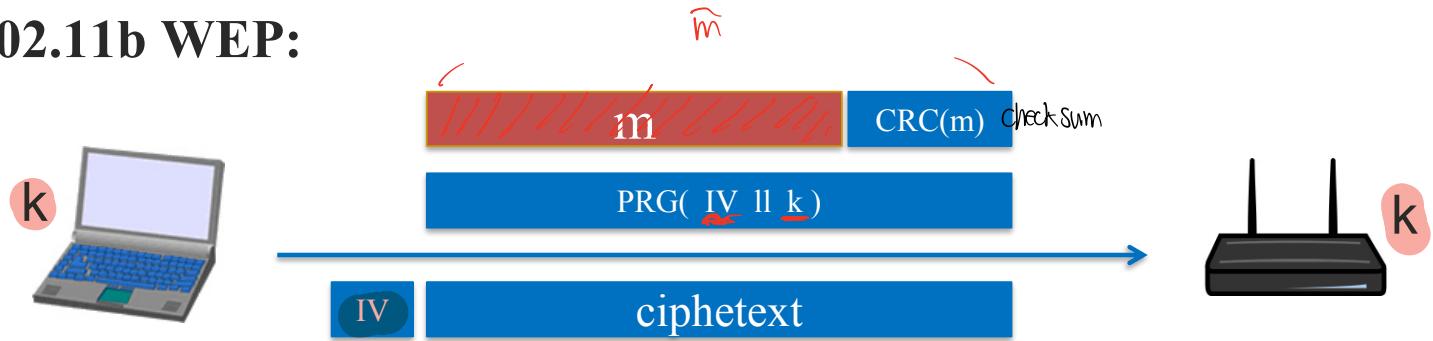


Need different keys for $C \rightarrow S$ and $S \rightarrow C$

Real Example of Stream Cipher (2)

PRG \Rightarrow RC4

802.11b WEP:



Length of IV: 24 bits

- Repeated IV after $2^{24} \approx 16M$ frames
- On some 802.11 cards: IV resets to 0 after power cycle
 - Key for frame #1: $(1||k)$
 - Key for frame #2: $(2||k)$

2^{24} 속도로 IV 사용 \Rightarrow two time pad
WEP의 초기화 문제에 허점이 있다.

2회 WEP
WEP2 사용.

eStream Project

- To identify new stream cipher (efficient and compact) suitable for widespread adoption
- Organized by the EU ECRTPT network
- Running from 2004 to 2008 and updated in 2009 and 2012
- Dedicated for unsynchronized mode PAGCK, IV
- eStream finalist

- HC-128
- Rabbit
- Sosemanuk
- Salsa20 현재 사용

SW-oriented

- MICKEY-128 v2
- Trivium 현재 사용 → LFSR-based
- Grain v1

HW-oriented

Modern Stream Ciphers: eStream

■ Unsynchronized mode (stateless way)

$$\text{PRG: } \underbrace{\{0,1\}^s}_{\text{seed}} \times R \rightarrow \{0,1\}^n$$

nonce *IV*

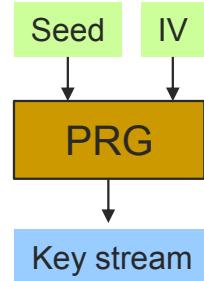
Nonce: a non-repeating value for a given key

(number used at once) 한 번 쓰는 random = 한 번 쓰는 숫자.
random이 아님에 고려해, 한 번 쓰는 숫자를 막함.

$$E(k, m; r) = m \oplus \text{PRG}(k; r)$$

- The pair (k,r) is never used more than once
- What is the advantage of using nonce?

정해진 규칙하에서 두 사용이 이루어지면 헛수고
정확하게 정해놓은 굳이 모든 발생이 둘다 같아야 선성 가능.

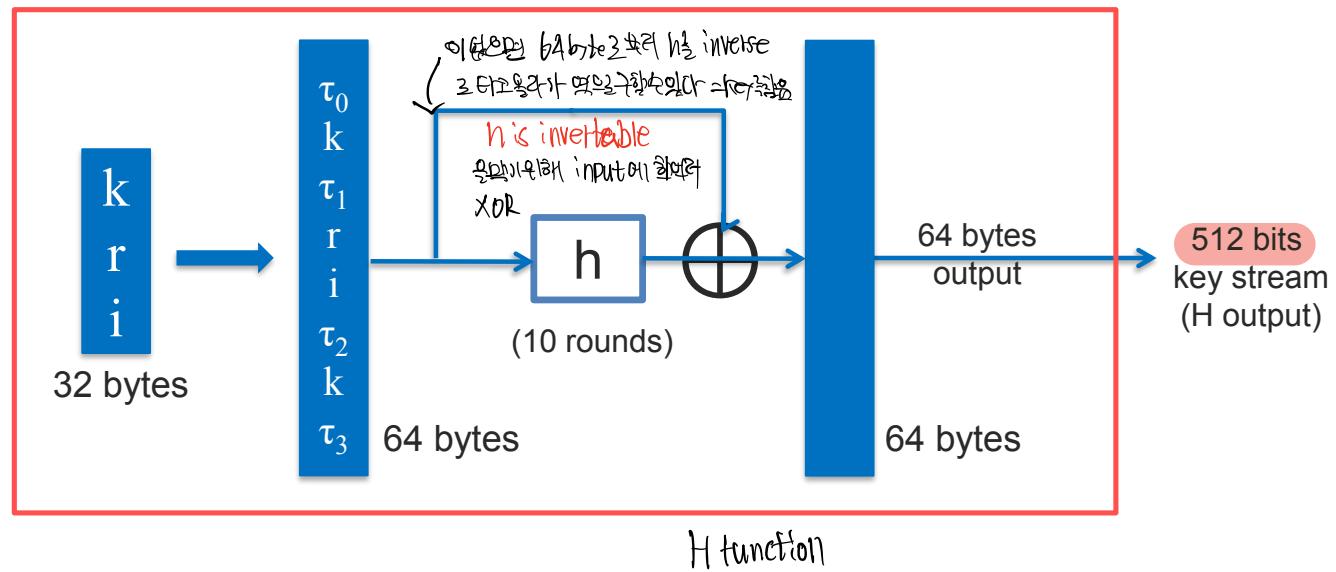


- o random #
 $\rightarrow r_1, r_2, r_3$
정수들 = uniform
- o nonce
 $\rightarrow 0, 1, 2, 3, 4, \dots$
정수들 구간마다 일관성이.

eStream: Salsa20 (SW or HW)

Salsa20: $\{0,1\}^{128 \text{ or } 256} \times \underbrace{\{0,1\}^{64}}_{\text{key } k} \xrightarrow{\text{nonce}} \underbrace{\{0,1\}^n}_{\text{(max } n = 2^{73} \text{ bits)}}$

Salsa20($k ; r$) := $H(k, (r, 0)) \parallel H(k, (r, 1)) \parallel \dots$



h : invertible function. designed to be fast on x86

Performance

AMD Opteron, 2.2 GHz (Linux)

	<u>PRG</u>	<u>Speed (MB/sec)</u>
	RC4	126
eStream	Salsa20/12	643
	Sosemanuk	727

Q & A